What I did to Adol-C and ISSM

Utke/Larour

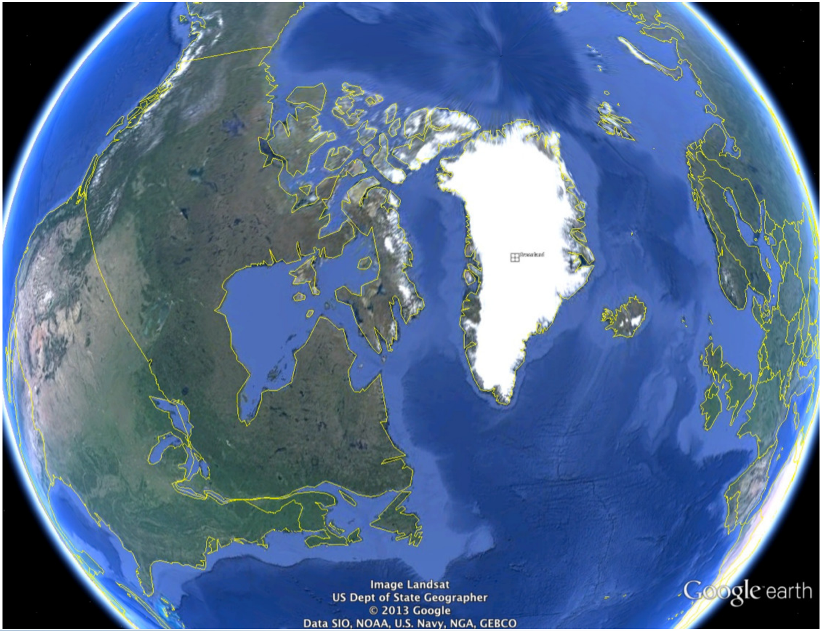
Argonne National Laboratory
Jet Propulsion Laboratory

EuroAD - Dec/2013 Oxford, UK

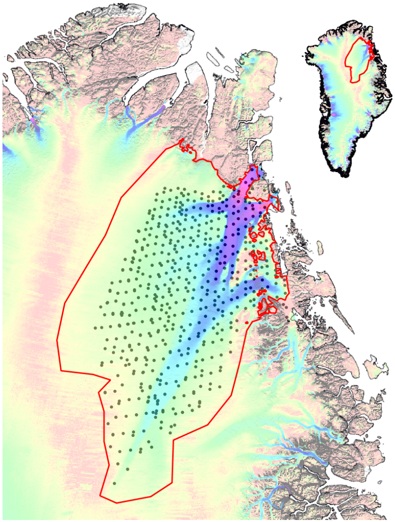
outline

- what this is for
- changes in ISSM
- changes in Adol-C
- external solvers
- adjoinable MPI
- performance

Greenland

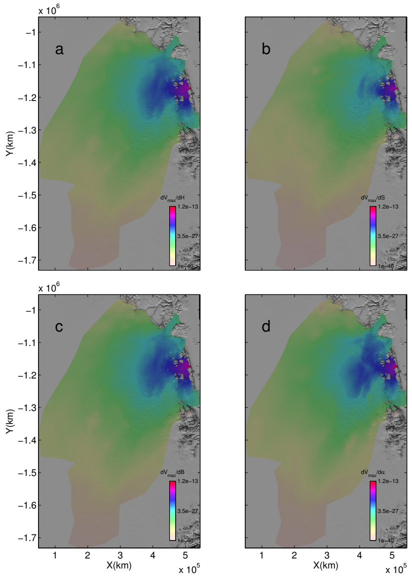

Greenland
Image Landsat
US Dept of State Geographer
© 2013 Google
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Google earth

the North-Eastern Ice Stream on Greenland



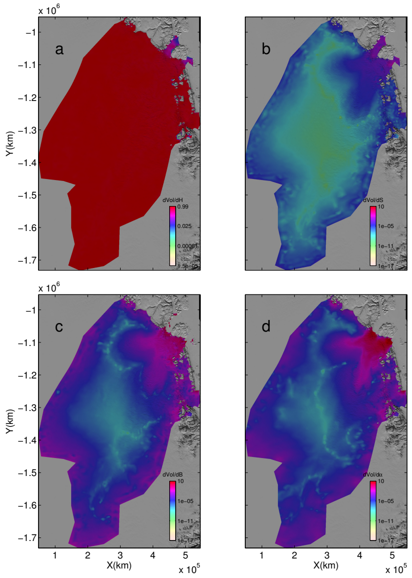
- velocity field
- red boundary shows domain of interest
- dots indicate observation data
- surface observations by satellite/stations
- drilling holes is expensive
- goal is model tuning for prediction of sea level rise

sensitivity studies - maximal velocity with respect to



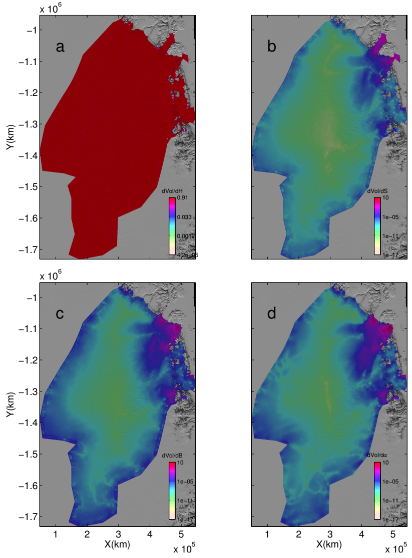
- a: H - ice thickness
- b: S - surface elevation
- c: B - bed elevation
- d: α - friction coefficient

sensitivity studies - volume with respect to



- a: H - ice thickness
- b: S - surface elevation
- c: B - bed elevation
- d: α - friction coefficient

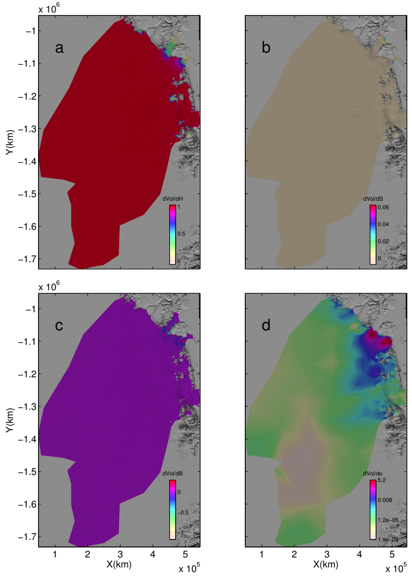
sensitivity studies (last week) - volume with respect to



- a: H - ice thickness
- b: S - surface elevation
- c: B - bed elevation
- d: α - friction coefficient

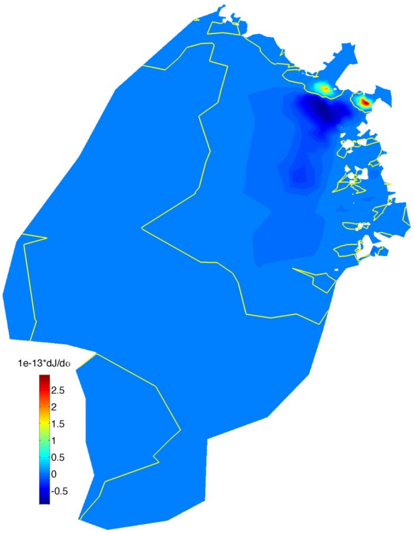
compared to earlier studies ran in higher resolution on Pleiades for longer model time

sensitivity studies - volume above floatation with respect to



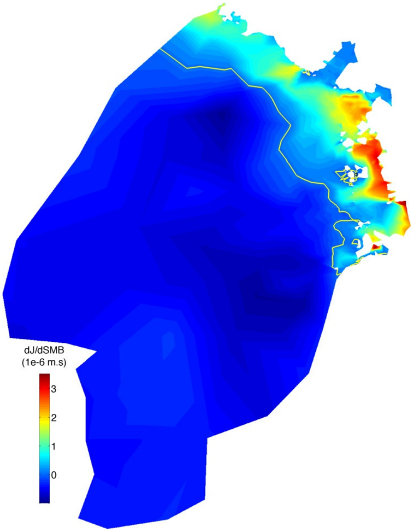
- a: H - ice thickness
- b: S - surface elevation
- c: B - bed elevation
- d: α - friction coefficient

model-to-observation misfit of S to internal state



- with respect to friction coefficients
- L^2 integrated over time
- yellow lines indicate gradient sign switch
- part of an gradient $\rightarrow$ line search optimization loop

model-to-observation misfit of S to external boundary



- with respect to snow-mass-balance
- snow fall given as (external) reanalysis of climate model runs
- hints at less snow fall on the coast, more inland
- means to adapt reanalysis if one assumes the ice sheet model is “correct”

presented at AGU meeting

ISSM

- C++ model

ISSM

- C++ model
- $\approx$ 750 files, 100K lines, 10 developers

ISSM

- C++ model
- ≈ 750 files, 100K lines, 10 developers
- uses templates, virtual methods, multiple inheritance (i.e. isn't just glorified C)

ISSM

- C++ model
- $\approx$ 750 files, 100K lines, 10 developers
- uses templates, virtual methods, multiple inheritance (i.e. isn't just glorified C)
- only the model core is C++, data pre/post processing done with Matlab / Python

ISSM

- C++ model
- $\approx$ 750 files, 100K lines, 10 developers
- uses templates, virtual methods, multiple inheritance (i.e. isn't just glorified C)
- only the model core is C++, data pre/post processing done with Matlab / Python
- parallelized with MPI

ISSM

- C++ model
- $\approx$ 750 files, 100K lines, 10 developers
- uses templates, virtual methods, multiple inheritance (i.e. isn't just glorified C)
- only the model core is C++, data pre/post processing done with Matlab / Python
- parallelized with MPI
- runs on *nix; NASA's Pleiades $\leftrightarrow$ Android

ISSM

- C++ model
- $\approx$ 750 files, 100K lines, 10 developers
- uses templates, virtual methods, multiple inheritance (i.e. isn't just glorified C)
- only the model core is C++, data pre/post processing done with Matlab / Python
- parallelized with MPI
- runs on *nix; NASA's Pleiades $\leftrightarrow$ Android
- has extensive regression testing (incl. the numerical results)

ISSM

- C++ model
- $\approx$ 750 files, 100K lines, 10 developers
- uses templates, virtual methods, multiple inheritance (i.e. isn't just glorified C)
- only the model core is C++, data pre/post processing done with Matlab / Python
- parallelized with MPI
- runs on *nix; NASA's Pleiades $\leftrightarrow$ Android
- has extensive regression testing (incl. the numerical results)
- uses libraries (meshing, partitioning, solvers)

AdolC-ify ISSM (1)

◇ `typedef` an `IssmDouble` and an `IssmPDouble` and switch on and off via `_HAVE_ADOLC_` configure define

AdolC-ify ISSM (1)

- `typedef` an `IssmDouble` and an `IssmPDouble` and switch on and off via `_HAVE_ADOLC_` configure define
- contributors deliver code in terms of `doubles` and expert developer categorizes those into `IssmDoubles` and `IssmPDoubles`
  $\Longrightarrow$ easy check

AdolC-ify ISSM (1)

- `typedef` an `IssmDouble` and an `IssmPDouble` and switch on and off via `_HAVE_ADOLC_` configure define
- contributors deliver code in terms of `doubles` and expert developer categorizes those into `IssmDoubles` and `IssmPDoubles`
  $\Longrightarrow$ easy check
- replace all `mallocs`, `news` and `frees`, `deletes` by templated `xNew` / `xDelete` incl. variants for 2-D arrays
  $\Longrightarrow$ safer, cleaner, more efficient; easy check

AdolC-ify ISSM (1)

- `typedef` an `IssmDouble` and an `IssmPDouble` and switch on and off via `_HAVE_ADOLC_` configure define
- contributors deliver code in terms of `doubles` and expert developer categorizes those into `IssmDoubles` and `IssmPDoubles`
  $\Longrightarrow$ easy check
- replace all `mallocs`, `news` and `frees`, `deletes` by templated `xNew` / `xDelete` incl. variants for 2-D arrays
  $\Longrightarrow$ safer, cleaner, more efficient; easy check
- templatize data containers (partially done)
- some undue activation (still) forced through Matlab interface

AdolC-ify ISSM (1)

- `typedef` an `IssmDouble` and an `IssmPDouble` and switch on and off via `_HAVE_ADOLC_` configure define
- contributors deliver code in terms of `doubles` and expert developer categorizes those into `IssmDoubles` and `IssmPDoubles`
  $\Longrightarrow$ easy check
- replace all `mallocs`, `news` and `frees`, `deletes` by templated `xNew` / `xDelete` incl. variants for 2-D arrays
  $\Longrightarrow$ safer, cleaner, more efficient; easy check
- templatize data containers (partially done)
- some undue activation (still) forced through Matlab interface
- passing data to passive code with templated `reCast`

AdolC-ify ISSM (1)

- `typedef` an `IssmDouble` and an `IssmPDouble` and switch on and off via `_HAVE_ADOLC_` configure define
- contributors deliver code in terms of `doubles` and expert developer categorizes those into `IssmDoubles` and `IssmPDoubles`
  $\Longrightarrow$ easy check
- replace all `mallocs`, `news` and `frees`, `deletes` by templated `xNew` / `xDelete` incl. variants for 2-D arrays
  $\Longrightarrow$ safer, cleaner, more efficient; easy check
- templatize data containers (partially done)
- some undue activation (still) forced through Matlab interface
- passing data to passive code with templated `reCast`
- `reCast` injections represent majority of the manual adaptation work

AdolC-ify ISSM (2)

- pick a simple(!) setup to start with and establish consistency with FD tests

AdolC-ify ISSM (2)

- pick a simple(!) setup to start with and establish consistency with FD tests
- few time steps, coarse resolution
- sequential, dense LU solve from GSL - needs wrapping

AdolC-ify ISSM (2)

- pick a simple(!) setup to start with and establish consistency with FD tests
- few time steps, coarse resolution
- sequential, dense LU solve from GSL - needs wrapping
- Adol-C has had an "external" interface for `func(x,y)` all along **but**:

AdolC-ify ISSM (2) & change Adol-C

- pick a simple(!) setup to start with and establish consistency with FD tests
- few time steps, coarse resolution
- sequential, dense LU solve from GSL - needs wrapping
- Adol-C has had an "external" interface for `func(x,y)` all along **but**:
    - the forward variants passed only $\dot{\mathtt{x}}$, $\dot{\mathtt{y}}$ and not `x`, `y` themselves $\Longrightarrow$ added

AdolC-ify ISSM (2) & change Adol-C

- pick a simple(!) setup to start with and establish consistency with FD tests
- few time steps, coarse resolution
- sequential, dense LU solve from GSL - needs wrapping
- Adol-C has had an "external" interface for `func(x,y)` all along **but**:
  - the forward variants passed only $\dot{x}$, $\dot{y}$ and not x, y themselves $\Longrightarrow$ added
  - added sanity checks for consecutive locations

AdolC-ify ISSM (2) & change Adol-C

- pick a simple(!) setup to start with and establish consistency with FD tests
- few time steps, coarse resolution
- sequential, dense LU solve from GSL - needs wrapping
- Adol-C has had an "external" interface for `func(x,y)` all along **but**:
  - the forward variants passed only $\dot{\mathtt{x}}$, $\dot{\mathtt{y}}$ and not `x`, `y` themselves $\Longrightarrow$ added
  - added sanity checks for consecutive locations
  - the forward/reverse handlers taped/restored `x`, `y` values whether needed or not $\Longrightarrow$ added controls

AdolC-ify ISSM (2) & change Adol-C

- pick a simple(!) setup to start with and establish consistency with FD tests
- few time steps, coarse resolution
- sequential, dense LU solve from GSL - needs wrapping
- Adol-C has had an "external" interface for `func(x,y)` all along **but**:
  - the forward variants passed only $\dot{x}$, $\dot{y}$ and not `x`, `y` themselves $\Longrightarrow$ added
  - added sanity checks for consecutive locations
  - the forward/reverse handlers taped/restored `x`, `y` values whether needed or not
    $\Longrightarrow$ added controls
  - time stepping $\rightarrow$ multiple solves with adaptive meshing $\rightarrow$ changing system dimensions
    $\Longrightarrow$ added tracking for maximum dimensions for single allocation of reusable help buffers

AdolC-ify ISSM (2) & change Adol-C

- pick a simple(!) setup to start with and establish consistency with FD tests
- few time steps, coarse resolution
- sequential, dense LU solve from GSL - needs wrapping
- Adol-C has had an "external" interface for `func(x,y)` all along **but**:
  - the forward variants passed only $\dot{\mathtt{x}}$, $\dot{\mathtt{y}}$ and not `x`, `y` themselves $\Longrightarrow$ added
  - added sanity checks for consecutive locations
  - the forward/reverse handlers taped/restored `x`, `y` values whether needed or not $\Longrightarrow$ added controls
  - time stepping $\rightarrow$ multiple solves with adaptive meshing $\rightarrow$ changing system dimensions $\Longrightarrow$ added tracking for maximum dimensions for single allocation of reusable help buffers
- got first regression tests passing with matching forward & reverse derivatives

AdolC-ify ISSM (3)

◇ so far the triumph of the "mature" tool

AdolC-ify ISSM (3)

- so far the triumph of the "mature" tool
- but then we expanded test cases and ...
- first problem - wrong forward values computed:

AdolC-ify ISSM (3)

- so far the triumph of the "mature" tool
- but then we expanded test cases and ...
- first problem - wrong forward values computed:
  - originated with partial array initializer list like `double a[3]={1.0,2.0};`

AdolC-ify ISSM (3)

- so far the triumph of the "mature" tool
- but then we expanded test cases and ...
- first problem - wrong forward values computed:
  - originated with partial array initializer list like `double a[3]={1.0,2.0};`
  - forces initialization in `adouble` default constructor

AdolC-ify ISSM (3)

- so far the triumph of the "mature" tool
- but then we expanded test cases and ...
- first problem - wrong forward values computed:
  - originated with partial array initializer list like `double a[3]={1.0,2.0};`
  - forces initialization in `adouble` default constructor
  - can be disabled in Adol-C configuration

AdolC-ify ISSM (3)

- so far the triumph of the "mature" tool
- but then we expanded test cases and ...
- first problem - wrong forward values computed:
  - originated with partial array initializer list like `double a[3]={1.0,2.0};`
  - forces initialization in `adouble` default constructor
  - can be disabled in Adol-C configuration
  - also changed in Rapsodia

AdolC-ify ISSM (3)

- so far the triumph of the "mature" tool
- but then we expanded test cases and ...
- first problem - wrong forward values computed:
    - originated with partial array initializer list like `double a[3]={1.0,2.0};`
    - forces initialization in `adouble` default constructor
    - can be disabled in Adol-C configuration
    - also changed in Rapsodia
- second problem - wrong forward values computed:

AdolC-ify ISSM (3)

- so far the triumph of the "mature" tool
- but then we expanded test cases and ...
- first problem - wrong forward values computed:
  - originated with partial array initializer list like `double a[3]={1.0,2.0};`
  - forces initialization in `adouble` default constructor
  - can be disabled in Adol-C configuration
  - also changed in Rapsodia
- second problem - wrong forward values computed:
  - originated with more recent location management

AdolC-ify ISSM (3)

- so far the triumph of the "mature" tool
- but then we expanded test cases and ...
- first problem - wrong forward values computed:
  - originated with partial array initializer list like `double a[3]={1.0,2.0};`
  - forces initialization in `adouble` default constructor
  - can be disabled in Adol-C configuration
  - also changed in Rapsodia
- second problem - wrong forward values computed:
  - originated with more recent location management
  - set of fixes (partially by my, partially by KK)

AdolC-ify ISSM (3)

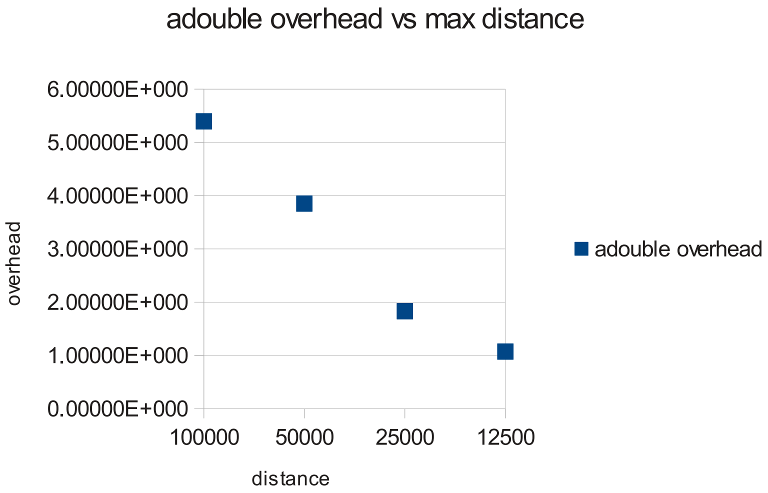
- so far the triumph of the "mature" tool
- but then we expanded test cases and ...
- first problem - wrong forward values computed:
  - originated with partial array initializer list like `double a[3]={1.0,2.0};`
  - forces initialization in `adouble` default constructor
  - can be disabled in Adol-C configuration
  - also changed in Rapsodia
- second problem - wrong forward values computed:
  - originated with more recent location management
  - set of fixes (partially by my, partially by KK)
- overhead?

AdolC-ified ISSM performance - overloading (1)

pick some test case (here "test109"), `g++ -O2`, initially horrible timings fixed by another changeset to locations mgmt. from KK

AdolC-ified ISSM performance - overloading (1)

pick some test case (here "test109"), `g++ -O2`, initially horrible timings fixed by another changeset to locations mgmt. from KK

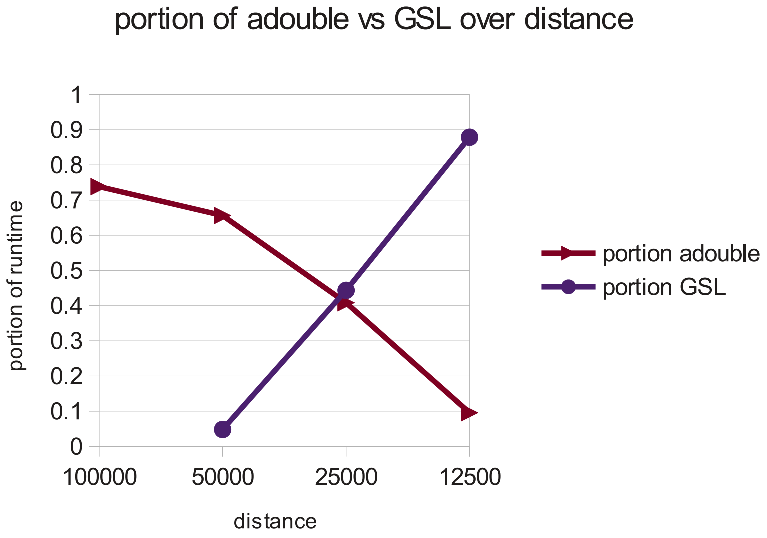


AdolC-ified ISSM performance - overloading (2)

less of a surprise once we look at the portions of runtime

AdolC-ified ISSM performance - overloading (2)

less of a surprise once we look at the portions of runtime



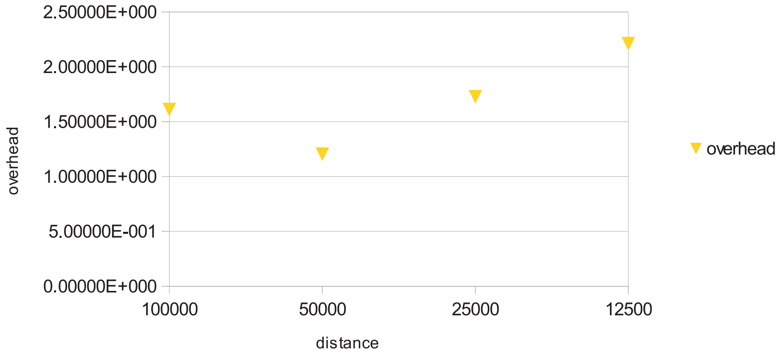
AdolC-ified ISSM performance - tracing & reverse (1)

using ”test3019” - an AD-enabled regression test

AdolC-ified ISSM performance - tracing & reverse (1)

using "test3019" - an AD-enabled regression test

overhead of tracing and fos_reverse over adouble run

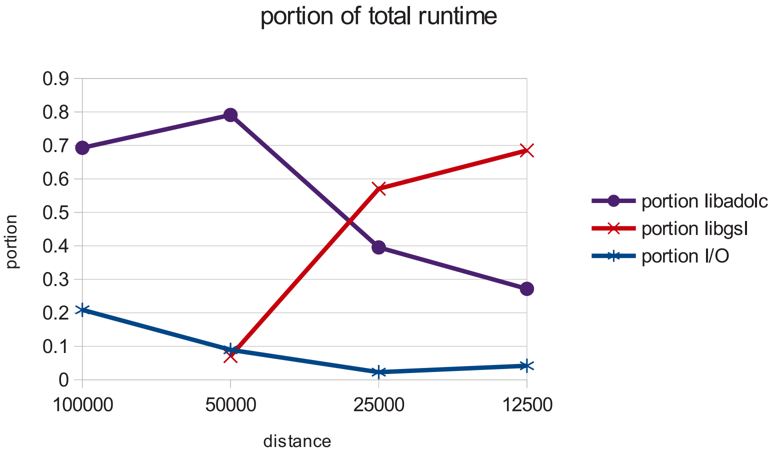


AdolC-ified ISSM performance - tracing & reverse (2)

using "test3019" - an AD-enabled regression test

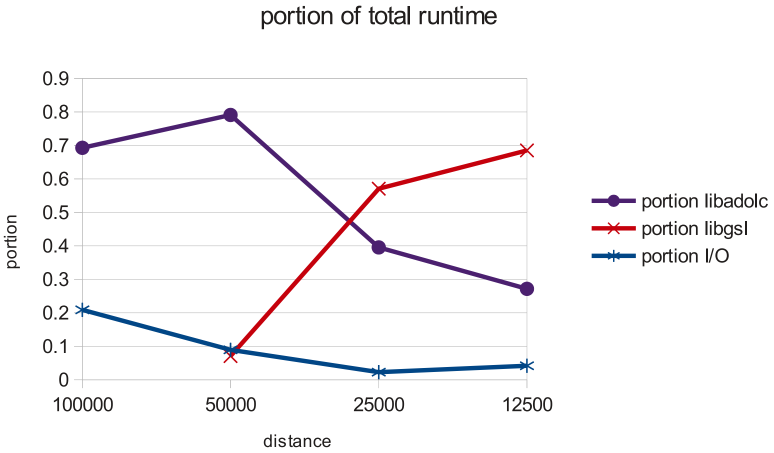
AdolC-ified ISSM performance - tracing & reverse (2)

using "test3019" - an AD-enabled regression test



AdolC-ified ISSM performance - tracing & reverse (2)

using "test3019" - an AD-enabled regression test



heavily skewed in Adol-C's advantage because of GSL

departing from simple setup

- different solver - MUMPS

departing from simple setup

- different solver - MUMPS
- sparse system

departing from simple setup

- different solver - MUMPS
- sparse system
- parallel setup

departing from simple setup

- different solver - MUMPS
- sparse system
- parallel setup
- had been working on Adjoinable MPI lib (w Laurent & Michel)

departing from simple setup

- different solver - MUMPS
- sparse system
- parallel setup
- had been working on Adjoinable MPI lib (w Laurent & Michel)
- most of the communication is collective which makes the adjoint "easier"; still needs the AMPI variant called

departing from simple setup

- different solver - MUMPS
- sparse system
- parallel setup
- had been working on Adjoinable MPI lib (w Laurent & Michel)
- most of the communication is collective which makes the adjoint "easier"; still needs the AMPI variant called
- don't like more preprocessor switches for using MPI or AMPI etc. littering the code

departing from simple setup

- different solver - MUMPS
- sparse system
- parallel setup
- had been working on Adjoinable MPI lib (w Laurent & Michel)
- most of the communication is collective which makes the adjoint "easier"; still needs the AMPI variant called
- don't like more preprocessor switches for using MPI or AMPI etc. littering the code
- introduce `ISSM_MPI` layer to encapsulate 4 versions

MPI	.	✓	.	✓
AD	.	.	✓	✓

departing from simple setup

- different solver - MUMPS
- sparse system
- parallel setup
- had been working on Adjoinable MPI lib (w Laurent & Michel)
- most of the communication is collective which makes the adjoint "easier"; still needs the AMPI variant called
- don't like more preprocessor switches for using MPI or AMPI etc. littering the code
- introduce `ISSM_MPI` layer to encapsulate 4 versions

MPI	.	✓	.	✓
AD	.	.	✓	✓

- fake MPI implementation uses `memcpy` or `adouble` assignments resp.

departing from simple setup

- different solver - MUMPS
- sparse system
- parallel setup
- had been working on Adjoinable MPI lib (w Laurent & Michel)
- most of the communication is collective which makes the adjoint "easier"; still needs the AMPI variant called
- don't like more preprocessor switches for using MPI or AMPI etc. littering the code
- introduce `ISSM_MPI` layer to encapsulate 4 versions

MPI	.	✓	.	✓
AD	.	.	✓	✓

- fake MPI implementation uses `memcpy` or `adouble` assignments resp.
- layer encapsulates all MPI switching $\Longrightarrow$ cleaner code

Adjoinable MPI in Adol-C and ISSM (1)

- Adol-C has:
  - (i) "tape"-based forward drivers
  - (ii) typed, 4-way stack (tape+Taylors)

Adjoinable MPI in Adol-C and ISSM (1)

- Adol-C has:
  - (i) "tape"-based forward drivers
  - (ii) typed, 4-way stack (tape+Taylors)

makes it *"pretty outstanding in terms of uniqueness"*

↰ Andreas' favorite quote of a Chicago tour guide

Adjoinable MPI in Adol-C and ISSM (1)

- Adol-C has:
  - (i) "tape"-based forward drivers
  - (ii) typed, 4-way stack (tape+Taylors)

  makes it *"pretty outstanding in terms of uniqueness"*

  ↰ Andreas' favorite quote of a Chicago tour guide
- (i) means using the same AMPI internal interface as Tapenade

Adjoinable MPI in Adol-C and ISSM (1)

- Adol-C has:
  - (i) "tape"-based forward drivers
  - (ii) typed, 4-way stack (tape+Taylors)

  makes it *"pretty outstanding in terms of uniqueness"*

  ↰ Andreas' favorite quote of a Chicago tour guide
- (i) means using the same AMPI internal interface as Tapenade
- (ii) reluctant to add a 5th stack for MPI parameters with opaque types (such as communicator, datatype etc.) $\Longrightarrow$ use Adj.-MPI provided default stack

Adjoinable MPI in Adol-C and ISSM (1)

- Adol-C has:
  - (i) "tape"-based forward drivers
  - (ii) typed, 4-way stack (tape+Taylors)

  makes it *"pretty outstanding in terms of uniqueness"*

  ↰ Andreas' favorite quote of a Chicago tour guide
- (i) means using the same AMPI internal interface as Tapenade
- (ii) reluctant to add a 5th stack for MPI parameters with opaque types (such as communicator, datatype etc.) $\Longrightarrow$ use Adj.-MPI provided default stack
- problem with the above is loss of self-containedness of the trace.

Adjoinable MPI in Adol-C and ISSM (1)

- Adol-C has:
  - (i) "tape"-based forward drivers
  - (ii) typed, 4-way stack (tape+Taylors)

  makes it *"pretty outstanding in terms of uniqueness"*

  ↰ Andreas' favorite quote of a Chicago tour guide
- (i) means using the same AMPI internal interface as Tapenade
- (ii) reluctant to add a 5th stack for MPI parameters with opaque types (such as communicator, datatype etc.) $\Longrightarrow$ use Adj.-MPI provided default stack
- problem with the above is loss of self-containedness of the trace.
- general problems with (re)storing blobs are related to (de)serialization of C++ objects

Adjoinable MPI in Adol-C and ISSM (1)

- Adol-C has:
  - (i) "tape"-based forward drivers
  - (ii) typed, 4-way stack (tape+Taylors)

  makes it *"pretty outstanding in terms of uniqueness"*

  ↰ Andreas' favorite quote of a Chicago tour guide
- (i) means using the same AMPI internal interface as Tapenade
- (ii) reluctant to add a 5th stack for MPI parameters with opaque types (such as communicator, datatype etc.) $\Longrightarrow$ use Adj.-MPI provided default stack
- problem with the above is loss of self-containedness of the trace.
- general problems with (re)storing blobs are related to (de)serialization of C++ objects
- covers all interfaces needed by ISSM (reductions, gather/scatter (v) combinations)

Adjoinable MPI in Adol-C and ISSM (1)

- Adol-C has:
  - (i) "tape"-based forward drivers
  - (ii) typed, 4-way stack (tape+Taylors)

  makes it *"pretty outstanding in terms of uniqueness"*
  ↰ Andreas' favorite quote of a Chicago tour guide
- (i) means using the same AMPI internal interface as Tapenade
- (ii) reluctant to add a 5th stack for MPI parameters with opaque types (such as communicator, datatype etc.) $\Longrightarrow$ use Adj.-MPI provided default stack
- problem with the above is loss of self-containedness of the trace.
- general problems with (re)storing blobs are related to (de)serialization of C++ objects
- covers all interfaces needed by ISSM (reductions, gather/scatter (v) combinations)
- deals with `MPI_INPLACE` and 0-count buffers

Adjoinable MPI in Adol-C and ISSM (1)

- Adol-C has:
  - (i) "tape"-based forward drivers
  - (ii) typed, 4-way stack (tape+Taylors)

  makes it *"pretty outstanding in terms of uniqueness"*

  ↰ Andreas' favorite quote of a Chicago tour guide
- (i) means using the same AMPI internal interface as Tapenade
- (ii) reluctant to add a 5th stack for MPI parameters with opaque types (such as communicator, datatype etc.) $\Longrightarrow$ use Adj.-MPI provided default stack
- problem with the above is loss of self-containedness of the trace.
- general problems with (re)storing blobs are related to (de)serialization of C++ objects
- covers all interfaces needed by ISSM (reductions, gather/scatter (v) combinations)
- deals with `MPI_INPLACE` and 0-count buffers
- requires contiguous locations $\Longrightarrow$ enforced in `xNew` spec.

Adjoinable MPI in Adol-C and ISSM (2)

introduction of "active" `AMPI_ADOUBLE` type to AMPI has been thoroughly discussed between the involved parties

Adjoinable MPI in Adol-C and ISSM (2)

introduction of "active" `AMPI_ADOUBLE` type to AMPI has been thoroughly discussed between the involved parties

◇ have coexisting active and passive communications

Adjoinable MPI in Adol-C and ISSM (2)

introduction of "active" `AMPI_ADOUBLE` type to AMPI has been thoroughly discussed between the involved parties

- have coexisting active and passive communications
- buffers passed by `void*`

Adjoinable MPI in Adol-C and ISSM (2)

introduction of "active" `AMPI_ADOUBLE` type to AMPI has been thoroughly discussed between the involved parties

- have coexisting active and passive communications
- buffers passed by `void*`
- forces some external distinction of activity

Adjoinable MPI in Adol-C and ISSM (2)

introduction of "active" `AMPI_ADOUBLE` type to AMPI has been thoroughly discussed between the involved parties

- ◇ have coexisting active and passive communications
- ◇ buffers passed by `void*`
- ◇ forces some external distinction of activity
- ◇ most natural by corresponding MPI types, particularly when one thinks of derived MPI types

Adjoinable MPI in Adol-C and ISSM (2)

introduction of "active" `AMPI_ADOUBLE` type to AMPI has been thoroughly discussed between the involved parties

- have coexisting active and passive communications
- buffers passed by `void*`
- forces some external distinction of activity
- most natural by corresponding MPI types, particularly when one thinks of derived MPI types
- the ISSM wrapper introduces `ISSM_MPI_DOUBLE` and `ISSM_MPI_PDOUBLE`

Adjoinable MPI in Adol-C and ISSM (2)

introduction of "active" AMPI_ADOUBLE type to AMPI has been thoroughly discussed between the involved parties

- have coexisting active and passive communications
- buffers passed by void*
- forces some external distinction of activity
- most natural by corresponding MPI types, particularly when one thinks of derived MPI types
- the ISSM wrapper introduces ISSM_MPI_DOUBLE and ISSM_MPI_PDOUBLE
- correspondence required the same way as in standard MPI

Adjoinable MPI in Adol-C and ISSM (2)

introduction of "active" AMPI_ADOUBLE type to AMPI has been thoroughly discussed between the involved parties

- have coexisting active and passive communications
- buffers passed by void*
- forces some external distinction of activity
- most natural by corresponding MPI types, particularly when one thinks of derived MPI types
- the ISSM wrapper introduces ISSM_MPI_DOUBLE and ISSM_MPI_PDOUBLE
- correspondence required the same way as in standard MPI
- practical problems with collectives distributed in the code proved it is easy to get wrong

Adjoinable MPI in Adol-C and ISSM (2)

template the MPI logic with 2 parameters like this pattern

```cpp
#include <iostream>
typedef int DataType;
class TypeInfo {
public:
  static DataType ourDoubleType;
  static DataType ourIntType;
};
DataType TypeInfo::ourDoubleType;
DataType TypeInfo::ourIntType;

template <class T, DataType *typeOfT_p> class C {
public:
  C(){};
  ~C(){};
  void foo(T aT) { std::cout << aT << "␣of␣type␣" << *typeOfT_p << std::endl; }
};

int main (void) {
  TypeInfo::ourDoubleType=1;
  TypeInfo::ourIntType=2;
  C<double,&TypeInfo::ourDoubleType>().foo(2.0);
  C<int,&TypeInfo::ourIntType>().foo(-1);
  return 0;
}
```

not completed (yet) in ISSM

wrapping MUMPS

- needs to convey sparsity info to wrapped solver

wrapping MUMPS

- needs to convey sparsity info to wrapped solver
- added integer array parameter to a second set of external func interfaces

wrapping MUMPS

- needs to convey sparsity info to wrapped solver
- added integer array parameter to a second set of external func interfaces
- suggests `void*` blobs again but has to address the same concerns as MPI opaque parameters

wrapping MUMPS

- needs to convey sparsity info to wrapped solver
- added integer array parameter to a second set of external func interfaces
- suggests `void*` blobs again but has to address the same concerns as MPI opaque parameters
- question whether to factorize again in the reverse sweep or recover factors:
    - generally - tradeoff fill-in for refactoring

wrapping MUMPS

- needs to convey sparsity info to wrapped solver
- added integer array parameter to a second set of external func interfaces
- suggests `void*` blobs again but has to address the same concerns as MPI opaque parameters
- question whether to factorize again in the reverse sweep or recover factors:
  - generally - tradeoff fill-in for refactoring
  - specifically - MUMPS can dump factors but is written in / geared toward Fortran

wrapping MUMPS

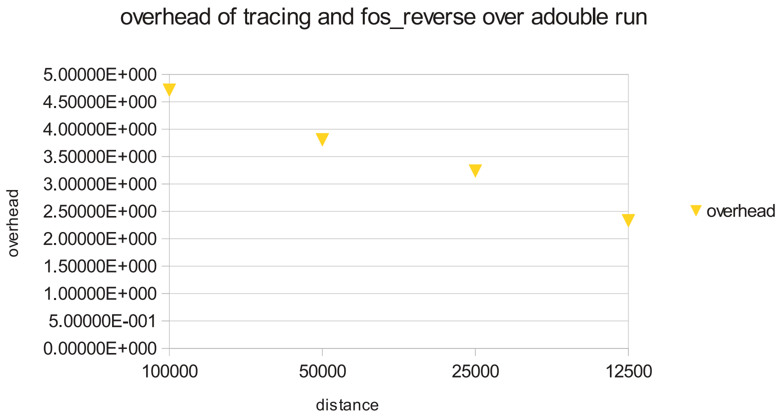
- needs to convey sparsity info to wrapped solver
- added integer array parameter to a second set of external func interfaces
- suggests `void*` blobs again but has to address the same concerns as MPI opaque parameters
- question whether to factorize again in the reverse sweep or recover factors:
  - generally - tradeoff fill-in for refactoring
  - specifically - MUMPS can dump factors but is written in / geared toward Fortran
  - performance analysis shows it is fast anyway (unlike GSL)

wrapping MUMPS

- needs to convey sparsity info to wrapped solver
- added integer array parameter to a second set of external func interfaces
- suggests `void*` blobs again but has to address the same concerns as MPI opaque parameters
- question whether to factorize again in the reverse sweep or recover factors:
  - generally - tradeoff fill-in for refactoring
  - specifically - MUMPS can dump factors but is written in / geared toward Fortran
  - performance analysis shows it is fast anyway (unlike GSL)
- revisit performance (by now $>$ 500 svn changesets later)

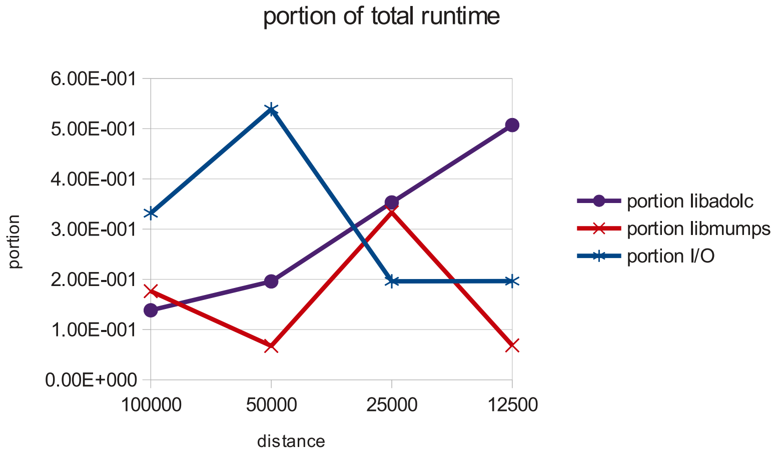
AdolC-ified ISSM performance - tracing & reverse (1)

test3019 - with contiguous locations, 3-way parallel MUMPS



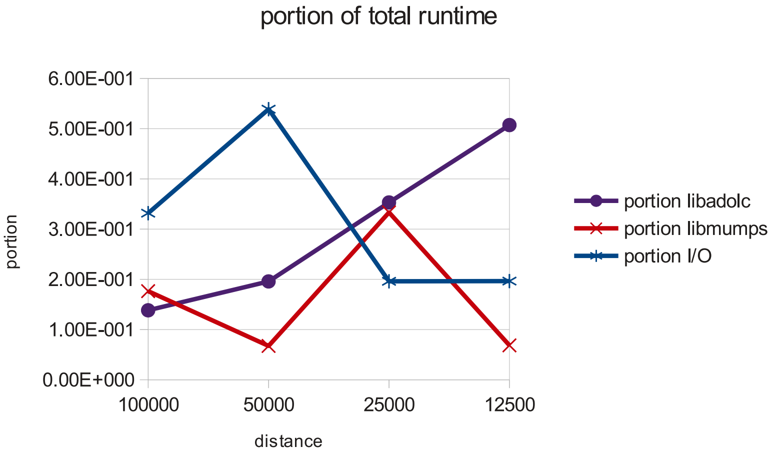
AdolC-ified ISSM performance - tracing & reverse (2)

test3019 - with contiguous locations, 3-way parallel MUMPS



AdolC-ified ISSM performance - tracing & reverse (2)

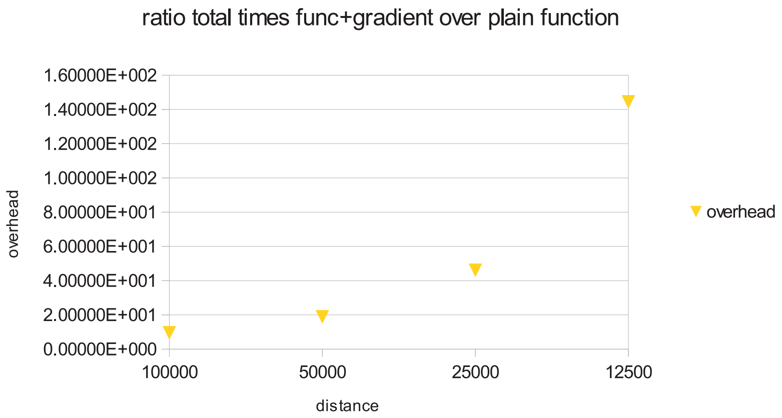
test3019 - with contiguous locations, 3-way parallel MUMPS



MUMPS is fast - more realistic picture

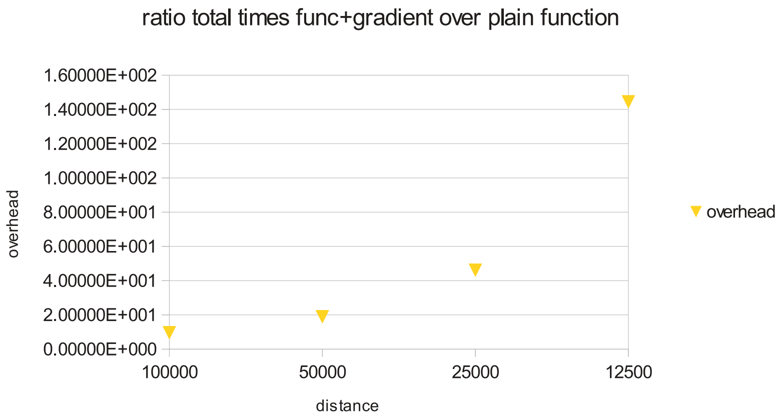
AdolC-ified ISSM performance - tracing & reverse (3)

test3019 - with contiguous locations, 3-way parallel MUMPS



AdolC-ified ISSM performance - tracing & reverse (3)

test3019 - with contiguous locations, 3-way parallel MUMPS



i.e. pretty nasty ... BUT

AdolC-ified ISSM performance - outlook

since these tests happened

◇ identified location management as one of the culprits
$\Longrightarrow$ added some logic to constrain the effort in consolidating the locations pool

AdolC-ified ISSM performance - outlook

since these tests happened

- ◇ identified location management as one of the culprits $\Longrightarrow$ added some logic to constrain the effort in consolidating the locations pool
- ◇ replaced dense (outside of MUMPS) matrix representation with a sparse format

AdolC-ified ISSM performance - outlook

since these tests happened

- ◇ identified location management as one of the culprits $\Longrightarrow$ added some logic to constrain the effort in consolidating the locations pool
- ◇ replaced dense (outside of MUMPS) matrix representation with a sparse format
- ◇ will need a new set of timing tests soon

AdolC-ified ISSM performance - outlook

since these tests happened

- identified location management as one of the culprits
  $\Longrightarrow$ added some logic to constrain the effort in consolidating the locations pool
- replaced dense (outside of MUMPS) matrix representation with a sparse format
- will need a new set of timing tests soon

current status:

AdolC-ified ISSM performance - outlook

since these tests happened

- ◇ identified location management as one of the culprits $\Longrightarrow$ added some logic to constrain the effort in consolidating the locations pool
- ◇ replaced dense (outside of MUMPS) matrix representation with a sparse format
- ◇ will need a new set of timing tests soon

current status:

- ◇ 60 - way runs on Pleiades

AdolC-ified ISSM performance - outlook

since these tests happened

- identified location management as one of the culprits $\Longrightarrow$ added some logic to constrain the effort in consolidating the locations pool
- replaced dense (outside of MUMPS) matrix representation with a sparse format
- will need a new set of timing tests soon

current status:

- 60 - way runs on Pleiades
- transient runs tax the file system and sometimes that causes crashes

AdolC-ified ISSM performance - outlook

since these tests happened

- identified location management as one of the culprits
  $\Longrightarrow$ added some logic to constrain the effort in consolidating the locations pool
- replaced dense (outside of MUMPS) matrix representation with a sparse format
- will need a new set of timing tests soon

current status:

- 60 - way runs on Pleiades
- transient runs tax the file system and sometimes that causes crashes
- acc. to Eric L.: **"You have to talk to it with love, and then you get numbers"**

AdolC-ified ISSM performance - outlook

since these tests happened

- ◇ identified location management as one of the culprits $\Longrightarrow$ added some logic to constrain the effort in consolidating the locations pool
- ◇ replaced dense (outside of MUMPS) matrix representation with a sparse format
- ◇ will need a new set of timing tests soon

current status:

- ◇ 60 - way runs on Pleiades
- ◇ transient runs tax the file system and sometimes that causes crashes
- ◇ acc. to Eric L.: **"You have to talk to it with love, and then you get numbers"**
- ◇ resilience/adjoint integrated checkpointing isn't there yet, but is in the works

AdolC-ified ISSM performance - outlook

since these tests happened

- ◇ identified location management as one of the culprits
  $\Longrightarrow$ added some logic to constrain the effort in consolidating the locations pool
- ◇ replaced dense (outside of MUMPS) matrix representation with a sparse format
- ◇ will need a new set of timing tests soon

current status:

- ◇ 60 - way runs on Pleiades
- ◇ transient runs tax the file system and sometimes that causes crashes
- ◇ acc. to Eric L.: **"You have to talk to it with love, and then you get numbers"**
- ◇ resilience/adjoint integrated checkpointing isn't there yet, but is in the works

IOW ... to be continued ...